

Geometric Transforms

김성영교수

금오공과대학교

컴퓨터공학과

학습 내용

- Geometric Transforms 개요
- Spatial transform
- Gray-level interpolation

기하학적 변환의 개요(계속)



[그림 8-4] 이동 영상

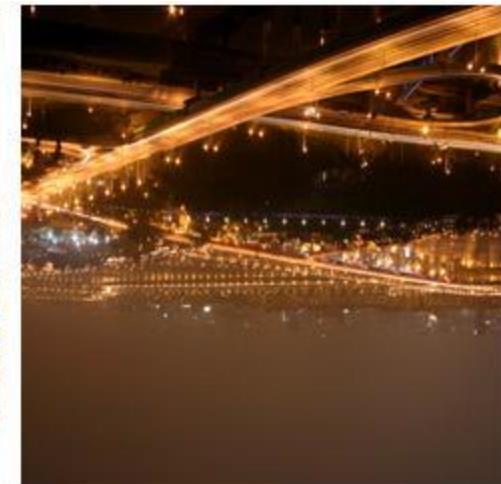
기하학적 변환의 개요(계속)



(a) 입력 영상

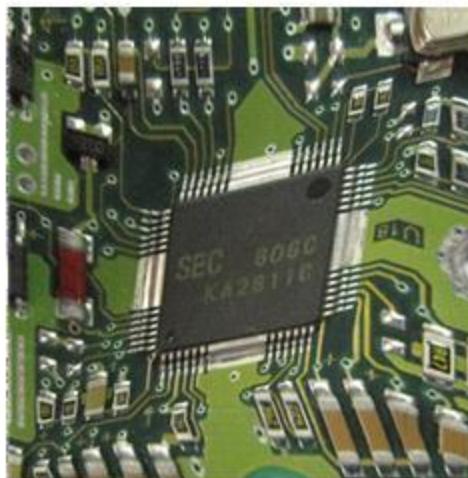


(b) 좌우 대칭 영상



(c) 상하 대칭 영상

[그림 8-5] 대칭 영상



[그림 8-6] 웜핑 처리 영상

기하학적 변환의 개요(계속)



(a) 입력 영상



(b) 중간 영상 단계 1



(c) 중간 영상 단계 2



(d) 중간 영상 단계 3



(e) 결과 영상

[그림 8-7] 모핑의 단계별 영상

Geometric Transforms 개요

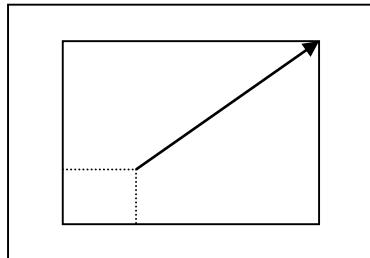
- 수식이나 변환 관계에 의해 픽셀들의 위치를 변경하는 변환
- 두 단계의 처리 단계로 구성
 - ① **mapping by spatial transform**
 - ② **gray-level interpolation**

$$\mathbf{D}(x, y) = \mathbf{I}(T_x(x, y), T_y(x, y))$$

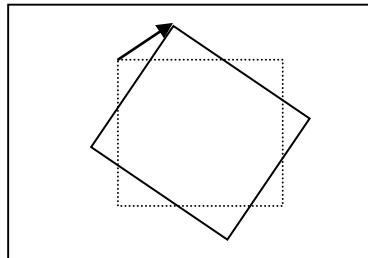
Spatial Transform

Map the input image location to a location in the output image

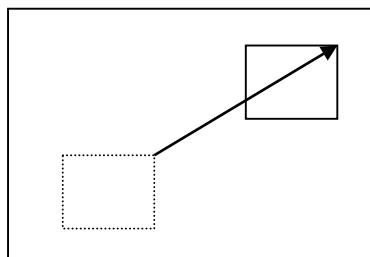
$$x' = T_x(x, y), \quad y' = T_y(x, y)$$



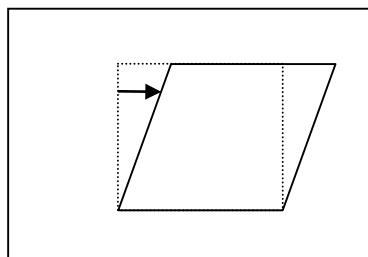
Scaling



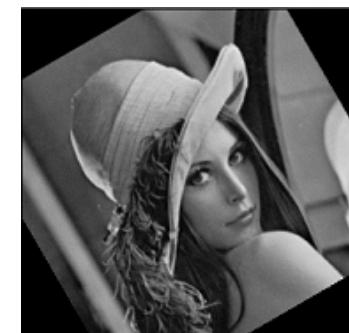
Rotation



Translation

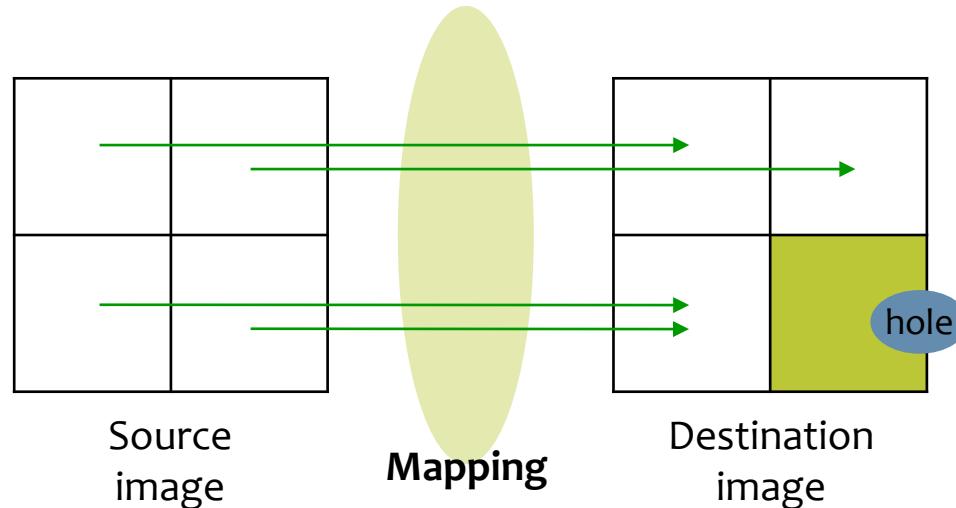


Skew



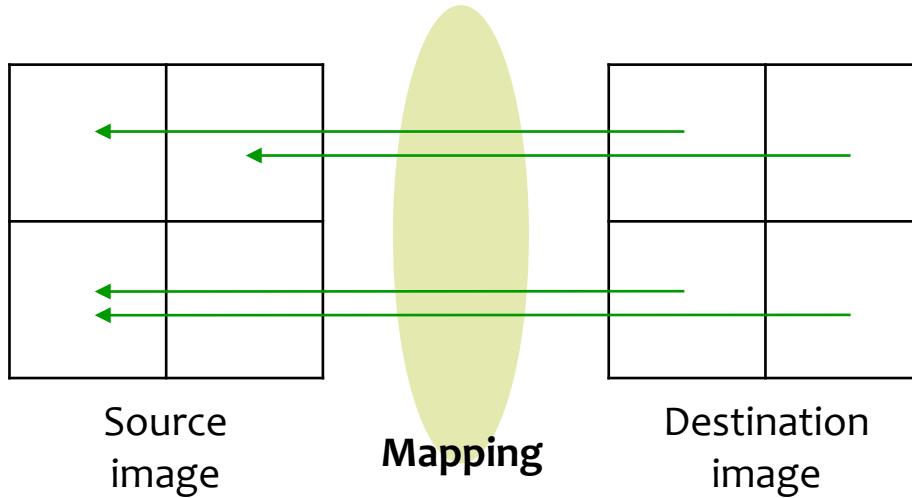
forward mapping

- 변환 수식에 의해 입력좌표를 출력좌표로 변환하는 과정
- 출력 영상에서 정의되지 않은 픽셀(hole) 발생



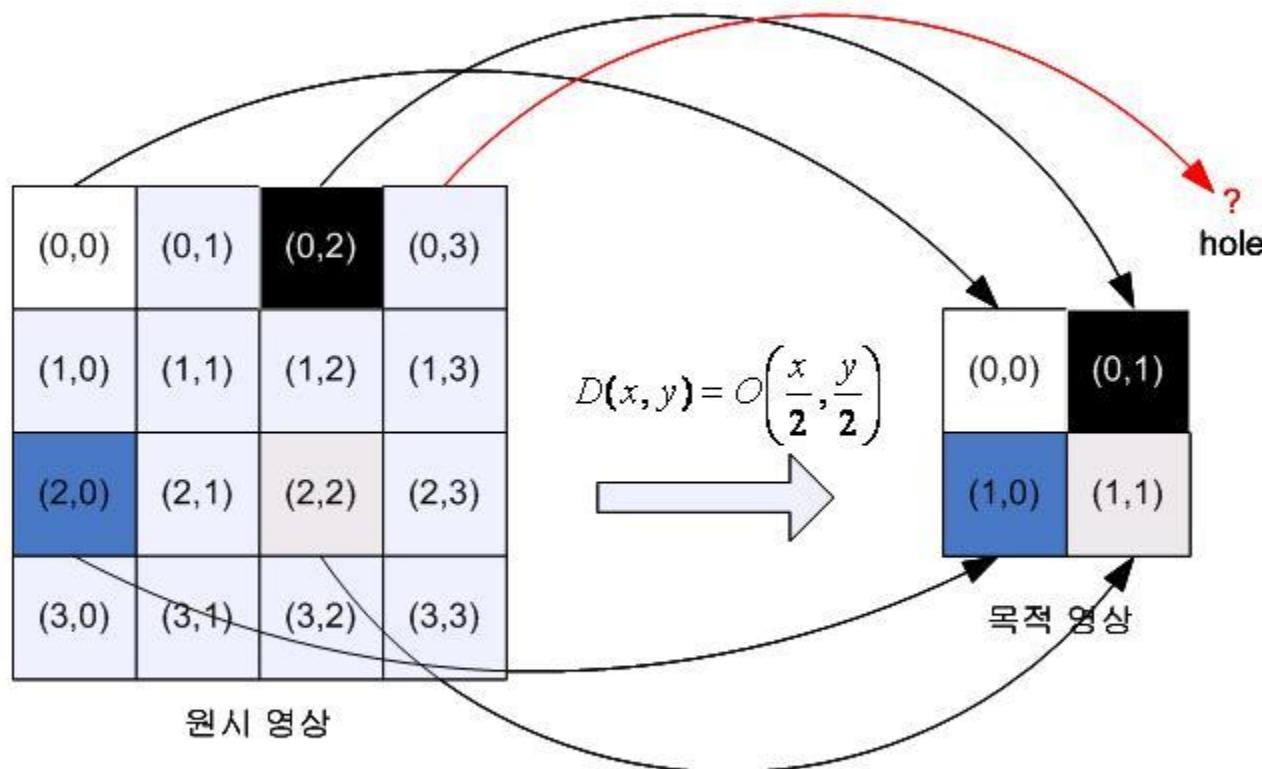
backward mapping

- 출력 영상의 각 픽셀 좌표에 대응하는 원본 영상의 좌표를 계산하여 해당 픽셀의 밝기 값을 결정하는 방법
- 출력 영상에서 정의되지 않은 픽셀 발생 방지
- 계산된 좌표가 정수가 아닌 경우 발생 → interpolation 적용



기하학 처리를 위한 전방향 사상과 역방향 사상 비교

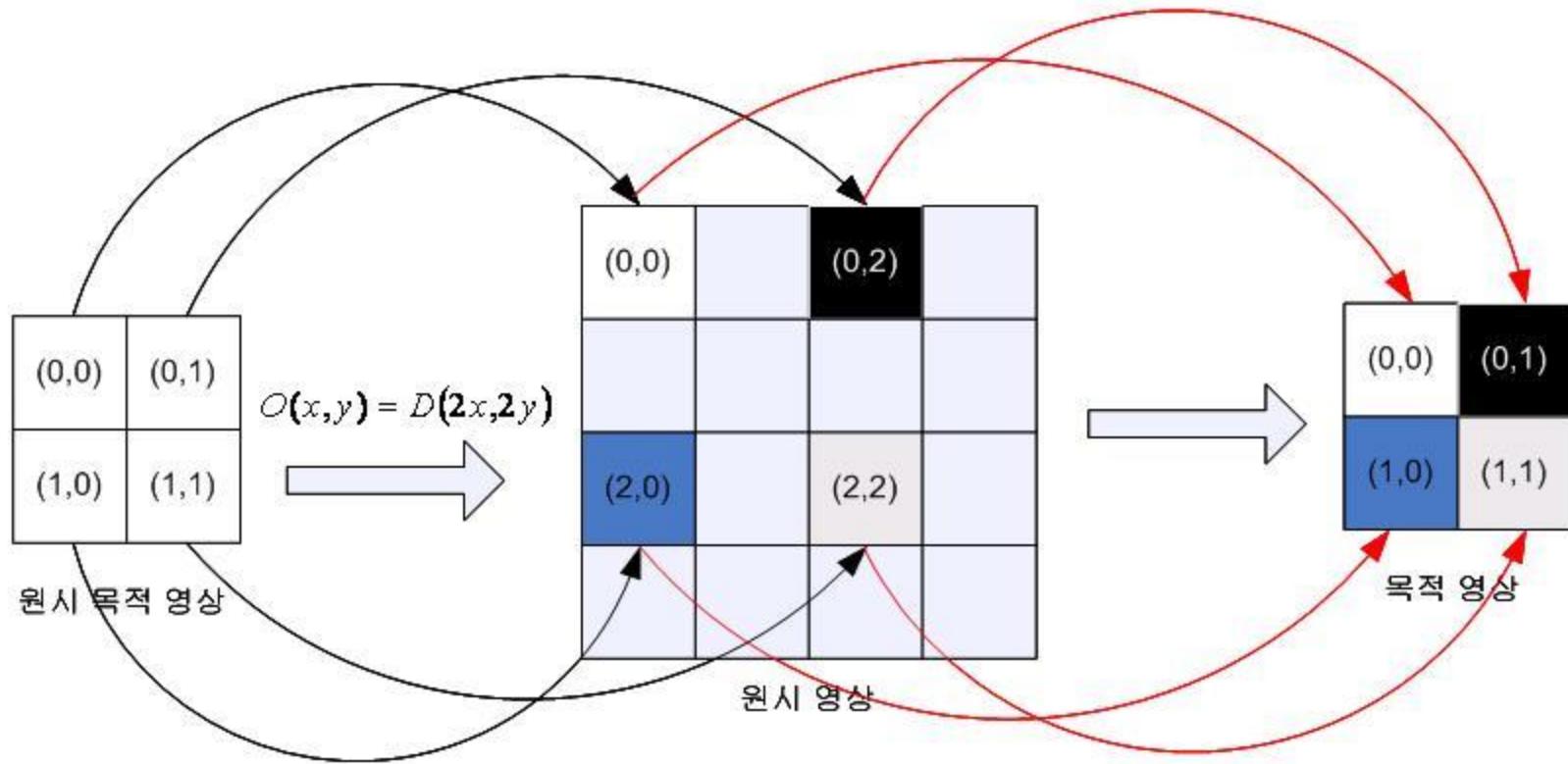
- ▶ 전방향 사상에서는 원시 영상의 좌표가 홀수면 목적 영상의 좌표 값에 소수점이 들어 있어 해당 좌표가 없게 되므로 홀 문제가 발생



[그림 8-11] 전방향 사상으로 생성된 목적 영상

기하학 처리를 위한 전방향 사상과 역방향 사상 비교(계속)

- 역함수로 원시 영상의 좌표 값을 계산하고, 그 좌표에 해당하는 화소 값을 찾아서 목적 영상에 할당
- 전방향 사상에서 발생했던 홀 문제가 일어나지 않음.



[그림 8-12] 역방향 사상으로 생성된 목적 영상

forward vs. backward mapping

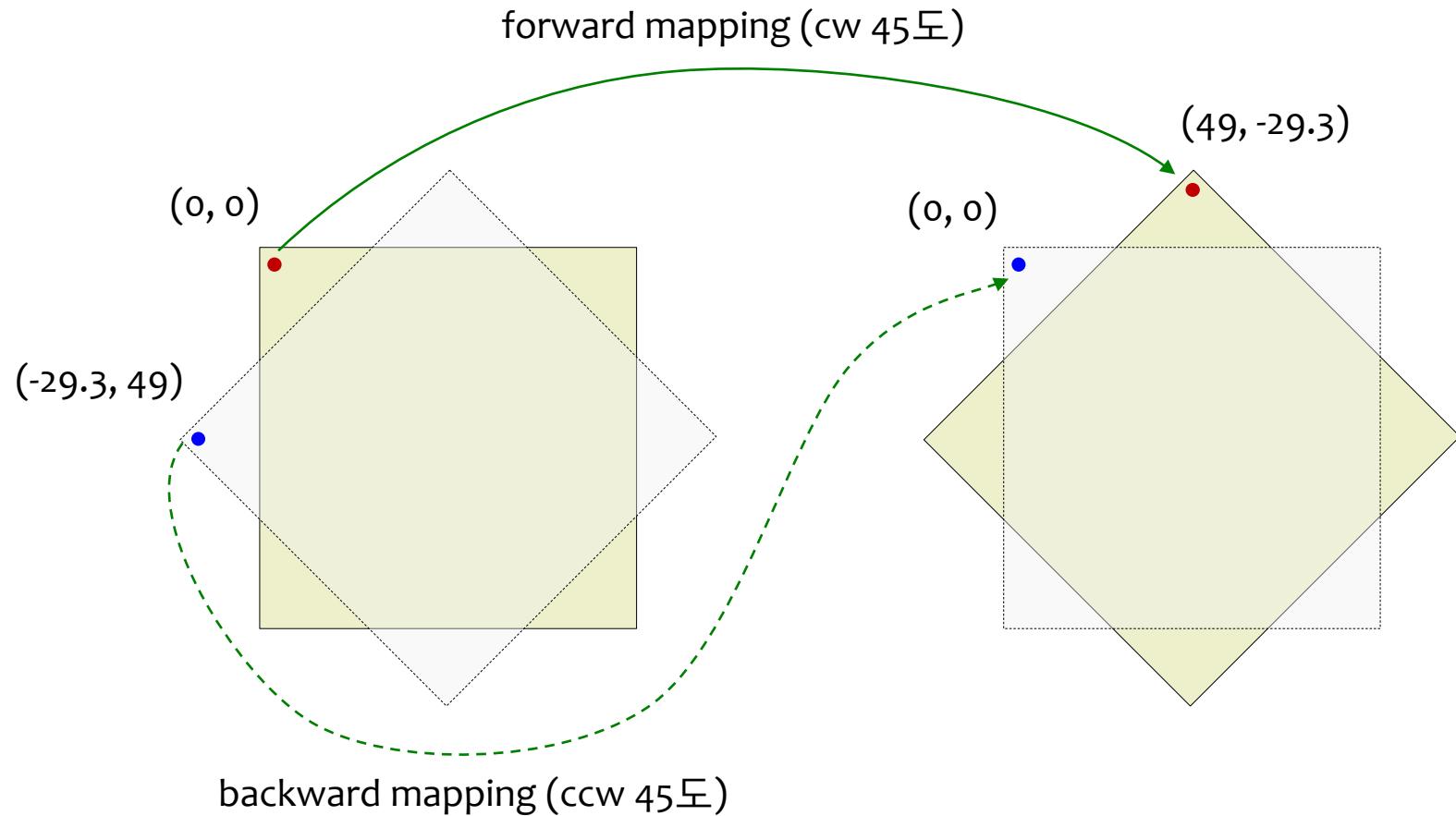


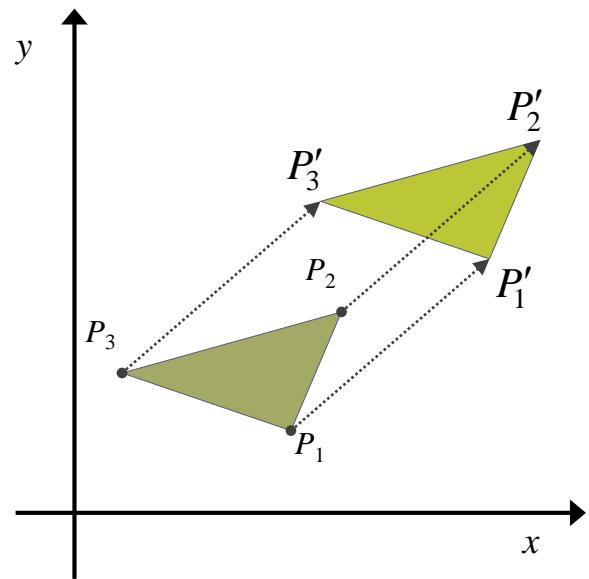
image size: 99 x 99, CW 45도 회전

Affine transform

- Linear transform
- 휘어짐이 없고 평행한 선들은 평행을 유지하는 변환
- 이동, 회전, 스케일 및 이들의 조합에 의한 변환
- $x' = a_0x + a_1y + a_2, \quad y' = b_0x + b_1y + b_2$
- homogeneous coordinate system 사용

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_0 & a_1 \\ b_0 & b_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_2 \\ b_2 \end{bmatrix}$$
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

translation



forward mapping

$$\begin{cases} x' = x + T_x \\ y' = y + T_y \end{cases} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

backward mapping

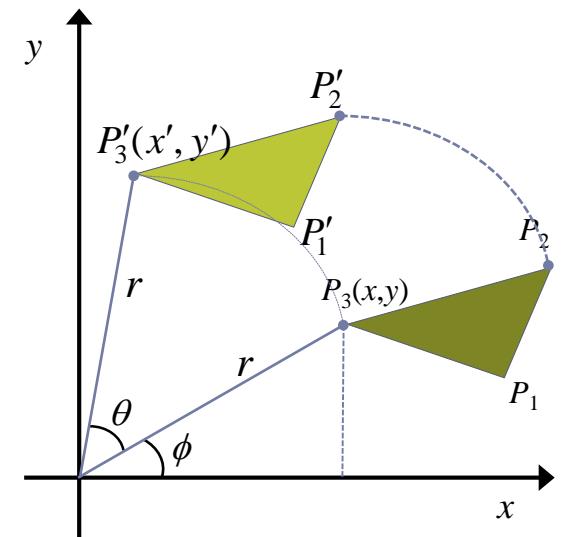
$$\begin{cases} x = x' - T_x \\ y = y' - T_y \end{cases} \quad \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -T_x \\ 0 & 1 & -T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

rotation

$$\begin{cases} x' = r \cdot \cos(\phi + \theta) = r \cdot \cos\phi \cos\theta - r \cdot \sin\phi \sin\theta = x \cos\theta - y \sin\theta \\ y' = r \cdot \sin(\phi + \theta) = r \cdot \sin\phi \cos\theta + r \cdot \cos\phi \sin\theta = x \sin\theta + y \cos\theta \end{cases}$$

$x = r \cdot \cos\phi, y = r \cdot \sin\phi$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \text{ forward mapping}$$

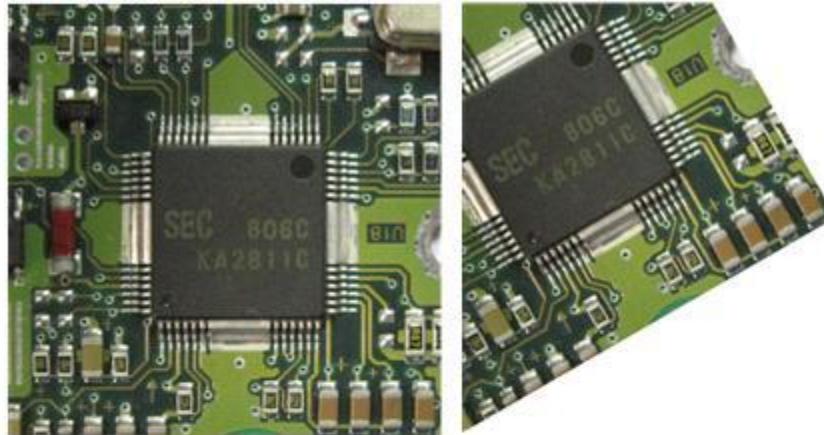


$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \text{ backward mapping}$$

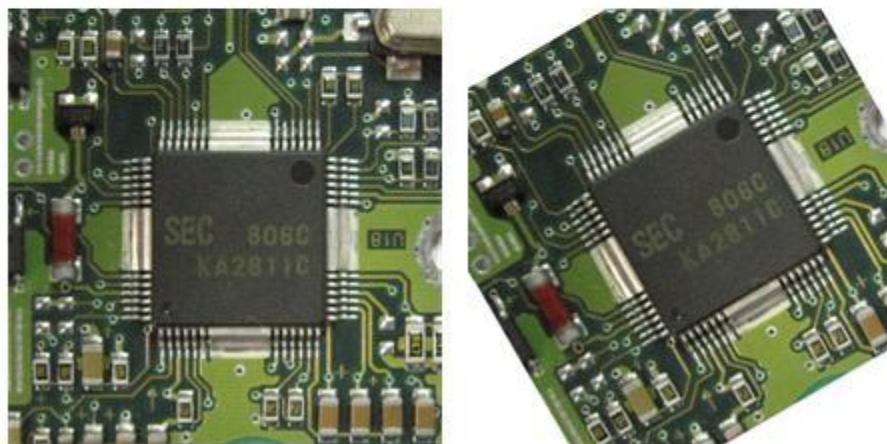
회전 결과 보이는 부분이 줄어드는 것을 방지하는 방법

▣ 회전 결과: 화소의 좌표 값이 음(-)

- 실제 화소 좌표는 음의 값일 수 없으므로 해당 부분은 잘려 안 보이게 됨.

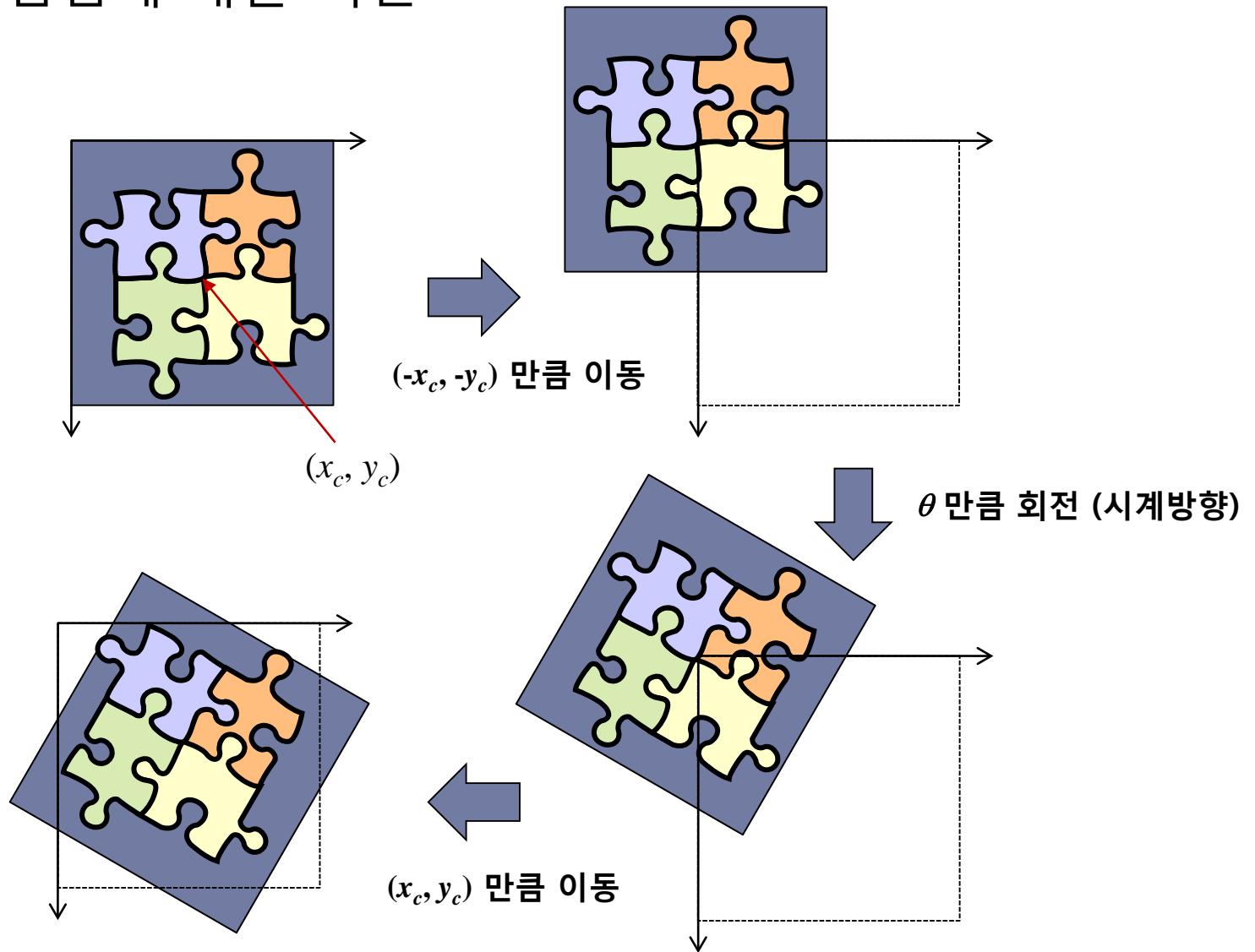


[그림 9-10] 원점을 기준으로 회전한 결과 영상



[그림 9-11] 영상의 중심점을 기준으로 회전한 결과 영상

영상 중심점에 대한 회전

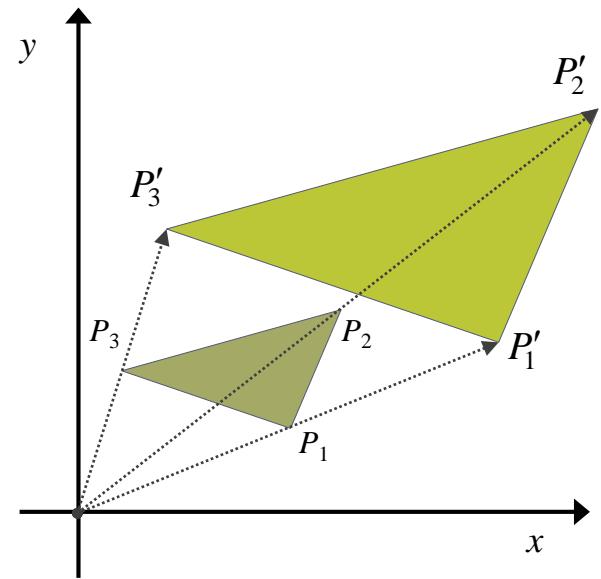


임의 점 (x, y) 에 대한 회전

- 1) 임의의 점을 원점으로 평행 위치 이동
- 2) 원점을 중심으로 회전
- 3) 단계 1)에서 수행한 이동의 반대 방향으로 평행 위치 이동

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -T_x \\ 0 & 1 & -T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} \cos \theta & -\sin \theta & T_x(1-\cos \theta) + T_y \sin \theta \\ \sin \theta & \cos \theta & T_y(1-\cos \theta) - T_x \sin \theta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

scaling



forward mapping

$$\begin{cases} x' = S_x \cdot x \\ y' = S_y \cdot y \end{cases}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

backward mapping

$$\begin{cases} x = S_x^{-1} \cdot x' \\ y = S_y^{-1} \cdot y' \end{cases}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} S_x^{-1} & 0 & 0 \\ 0 & S_y^{-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

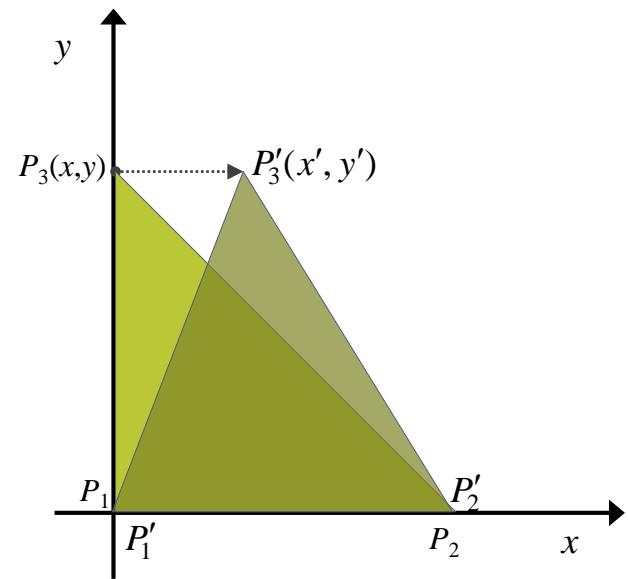
skew

forward mapping (about x)

$$\begin{cases} x' = x + u \cdot y \\ y' = y \end{cases} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & u & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

backward mapping (about x)

$$\begin{cases} x = x' - u \cdot y' \\ y = y' \end{cases} \quad \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & -u & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$



좌우 대칭

좌우 대칭

- 영상을 세로축을 중심으로 뒤집는 것
- 즉, 영상 내의 한 수직선을 중심으로 왼쪽 화소와 오른쪽 화소를 서로 교환하는 것

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -(x - x_0) \\ y \end{bmatrix} + \begin{bmatrix} x_0 \\ 0 \end{bmatrix}$$

(0.4)	(1.4)	(2.4)	(3.4)	(4.4)
(0.3)	(1.3)	(2.3)	(3.3)	(4.3)
(0.2)	(1.2)	(2.2)	(3.2)	(4.2)
(0.1)	(1.1)	(2.1)	(3.1)	(4.1)
(0.0)	(1.0)	(2.0)	(3.0)	(4.0)

(a) 원시 영상

(4.4)	(3.4)	(2.4)	(1.4)	(0.4)
(4.3)	(3.3)	(2.3)	(1.3)	(0.3)
(4.2)	(3.2)	(2.2)	(1.2)	(0.2)
(4.1)	(3.1)	(2.1)	(1.1)	(0.1)
(4.0)	(3.0)	(2.0)	(0.0)	(0.0)

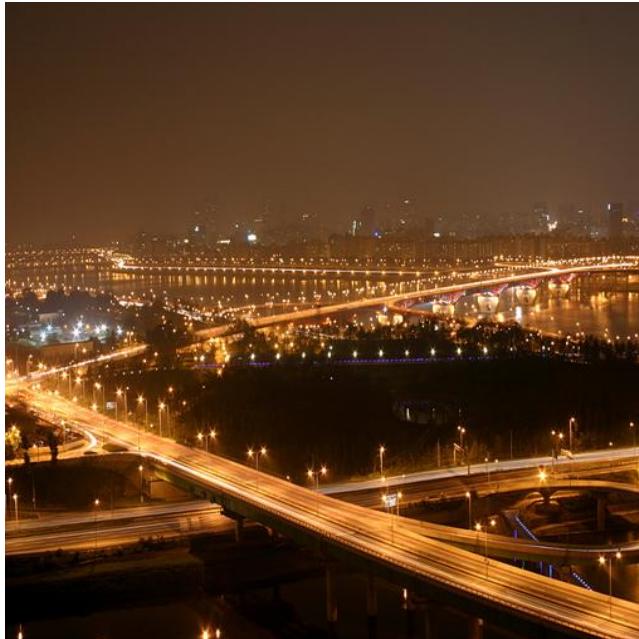
(b) 목적 영상

[그림 9-2] 이동 변환으로 화소 이동

[실습하기 9-2] 좌우 대칭 기하학 변환 프로그램

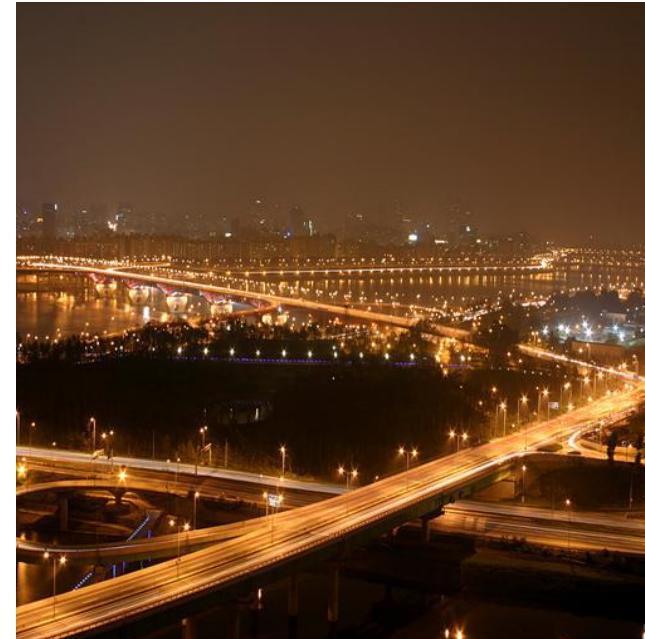
⑤ 프로그램 실행 결과 영상

- 거울에 비치는 효과와 같다고 해서 좌우 대칭을 거울 영상이라고 함.



(a) 입력 영상

좌우 대칭을 이용한 결과 영상



(b) 좌우 대칭 영상

상하 대칭

상하 대칭

- 영상을 가로축을 중심으로 뒤집는 것
- 즉, 영상 내의 한 수평선을 중심으로 위쪽의 화소와 아래쪽의 화소를 교환하는 것

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ -(y - y_0) \end{bmatrix} + \begin{bmatrix} 0 \\ y_0 \end{bmatrix}$$

(0,4)	(1,4)	(2,4)	(3,4)	(4,4)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)
(0,0)	(1,0)	(2,0)	(3,0)	(4,0)

(a) 원시 영상

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)
(0,4)	(1,4)	(2,4)	(3,4)	(4,4)

(b) 목적 영상

[그림 9-3] 좌우 대칭으로 화소 이동

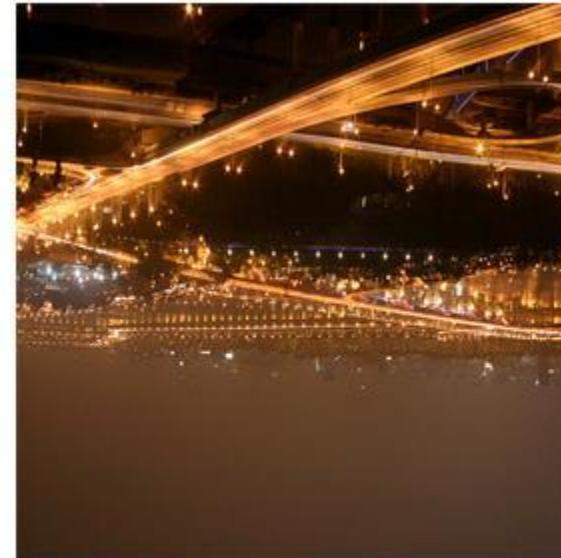
[실습하기 9-3] 상하 대칭 기하학 변환 프로그램

⑤ 프로그램 실행 결과 영상

- 입력 영상이 상하로 서로 뒤집힘.



(a) 입력 영상



(b) 상하 대칭 영상

상하 대칭을 이용한 결과 영상

요약

- Geometric Transforms

- 수식이나 변환 관계에 의해 픽셀들의 위치를 변경하는 변환

- Mapping by spatial transform

- 방식: forward 및 backward mapping
 - 종류: Affine transform 및 Warping (Perspective transform)

- Gray-level interpolation

- Nearest neighbor interpolation
 - Neighbor averaging interpolation
 - Bilinear interpolation

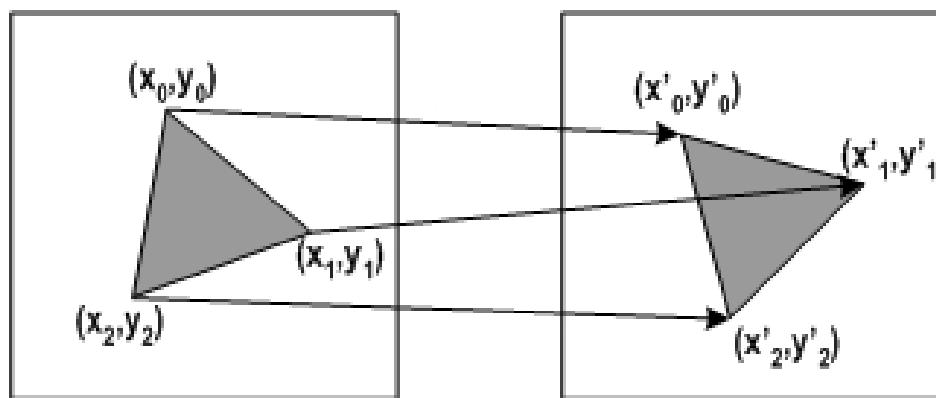
Reference

- R. Gonzalez, R. Woods, **Digital Image Processing (2nd Edition)**, Prentice Hall, 2002
- Scott E Umbaugh, **Computer Imaging**, CRC Press, 2005

To calculate an affine transform

- 6개의 미지수를 결정해야 함
- 3개의 좌표 점이 필요

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_o & a_1 & a_2 \\ b_o & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Warping

- Nonlinear transform (rubber sheet transform)
- pixel별로 이동 정도를 다르게 할 수 있어서 고무판 위에 그려진 영상을 임의대로 구부리는 것과 같은 효과를 낼 수 있음
- 고차항을 사용하여 일반화된 다항식으로 표현

$$x' = \sum_{i=0}^N \sum_{j=0}^{N-i} a_{ij} x^i y^j$$

$$y' = \sum_{i=0}^N \sum_{j=0}^{N-i} b_{ij} x^i y^j$$

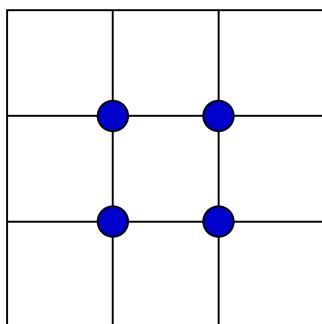
1st order warping

$$x' = \sum_{i=0}^N \sum_{j=0}^{N-i} a_{ij} x^i y^j$$

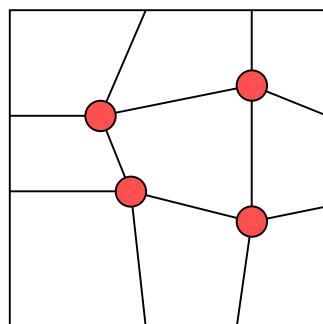
$$y' = \sum_{i=0}^N \sum_{j=0}^{N-i} b_{ij} x^i y^j$$

$$x' = a_0 x + a_1 y + a_2 xy + a_3,$$

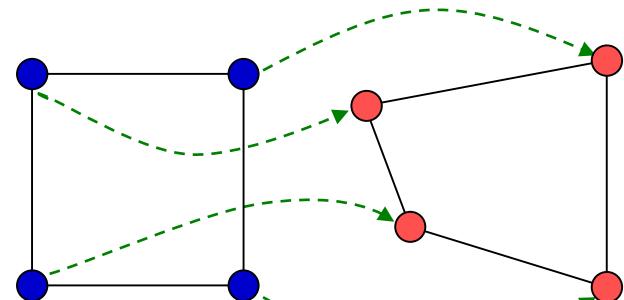
$$y' = b_0 x + b_1 y + b_2 xy + b_3$$



$$d(x', y')$$



$$I(x, y)$$





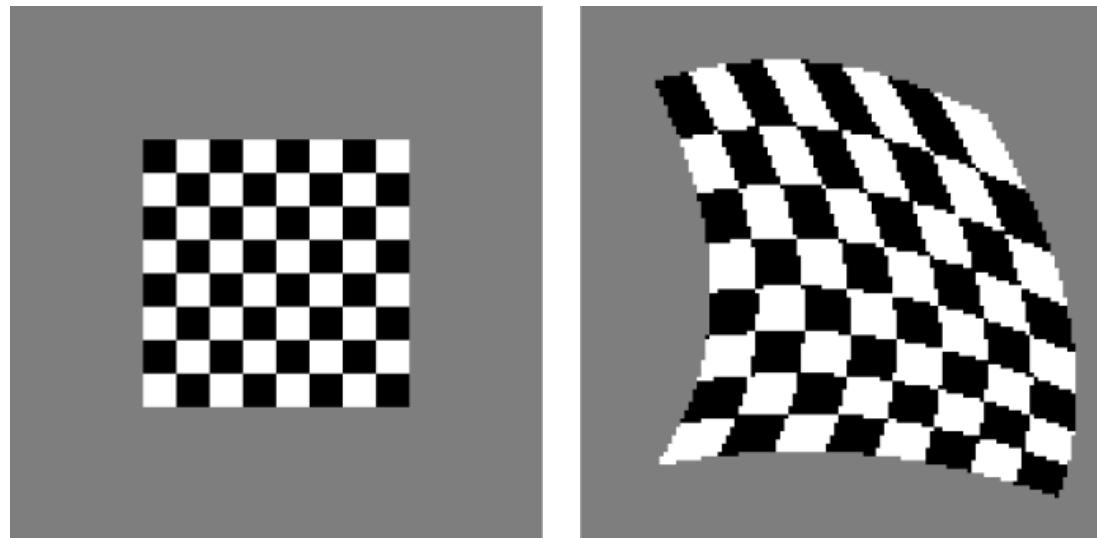
2nd order warping

$$x' = \sum_{i=0}^N \sum_{j=0}^{N-i} a_{ij} x^i y^j$$

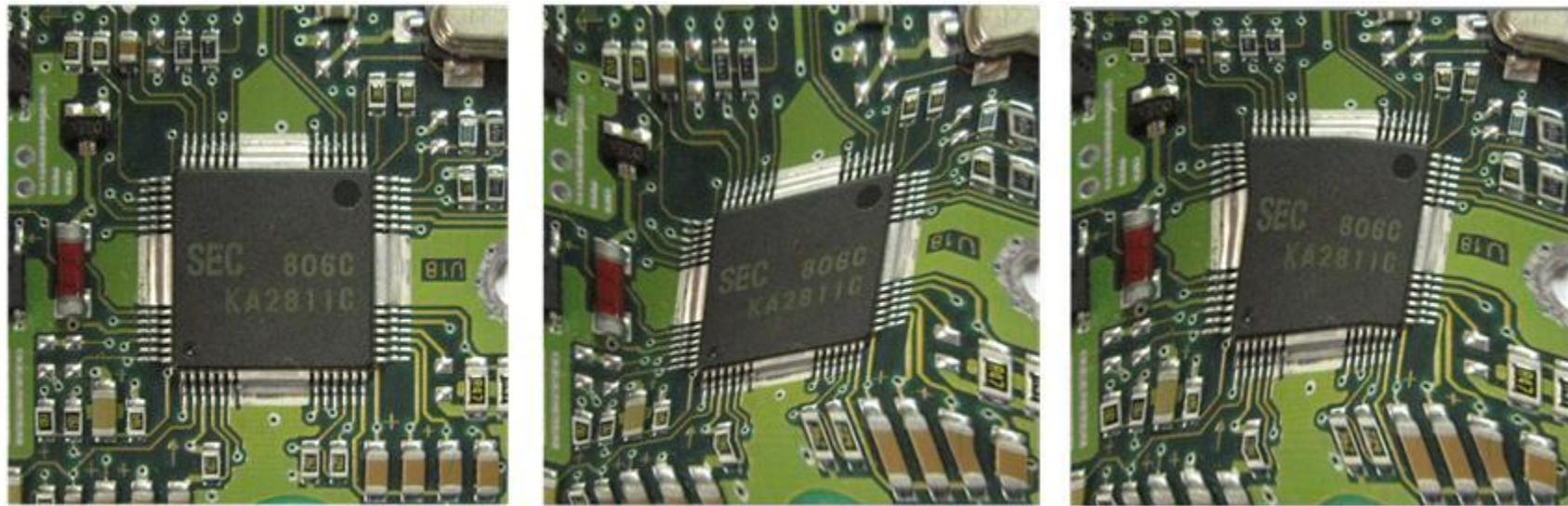
$$y' = \sum_{i=0}^N \sum_{j=0}^{N-i} b_{ij} x^i y^j$$

$$x' = a_0 x^2 + a_1 y^2 + a_2 xy + a_3 x + a_4 y + a_5,$$

$$y' = b_0 x^2 + b_1 y^2 + b_2 xy + b_3 x + b_4 y + b_5$$



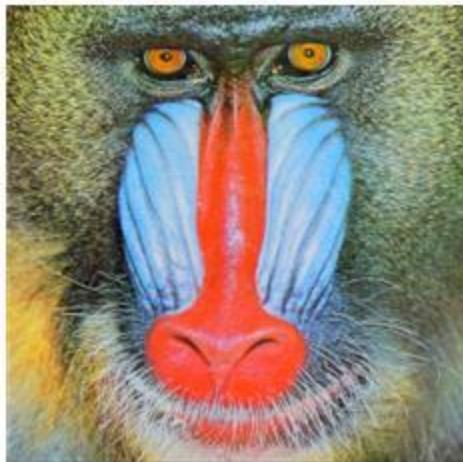
영상의 워핑 기하학적 변환[계속]



[그림 9-16] 워핑 처리된 결과 영상

모핑 기술

- 변형(Metamorphosis)에서 유래된 모핑(Morphing)은 한 영상을 서서히 변화시켜 다른 영상으로 변환하는 기술
- 원본 영상과 최종 영상은 물론, 최종 영상으로 매끄럽게 변할 수 있도록 많은 중간 단계의 영상도 필요함.



초기영상



중간영상



최종영상

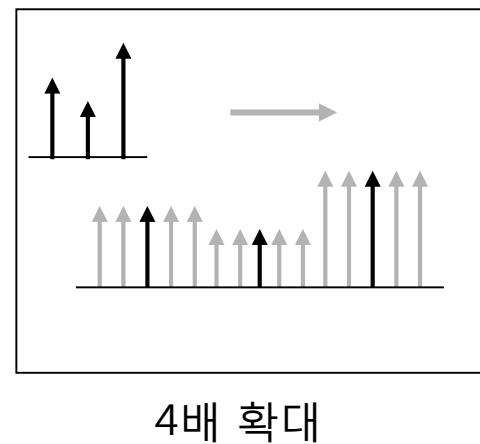
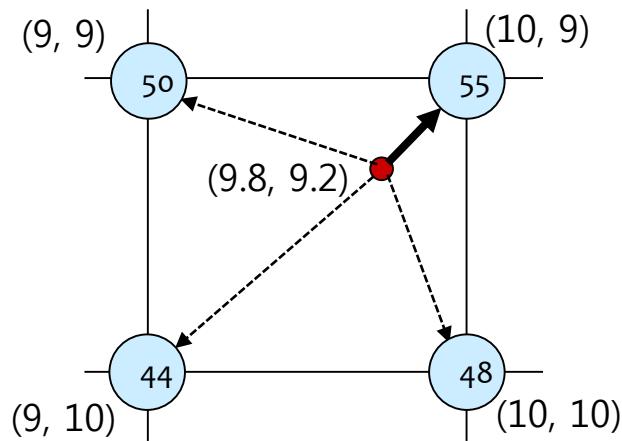
[그림 9-17] 중간 단계 영상을 생성하는 모핑의 수행 과정

Interpolation

- Interpolation 이란?
 - 결과 픽셀에 정확하게 대응되는 입력 픽셀이 없는 경우 주변 픽셀들을 고려하여 새로운 값을 생성하는 방법
- Interpolation 종류
 - Nearest neighbor interpolation
 - Neighbor averaging interpolation
 - Bilinear interpolation
 - Higher order interpolation

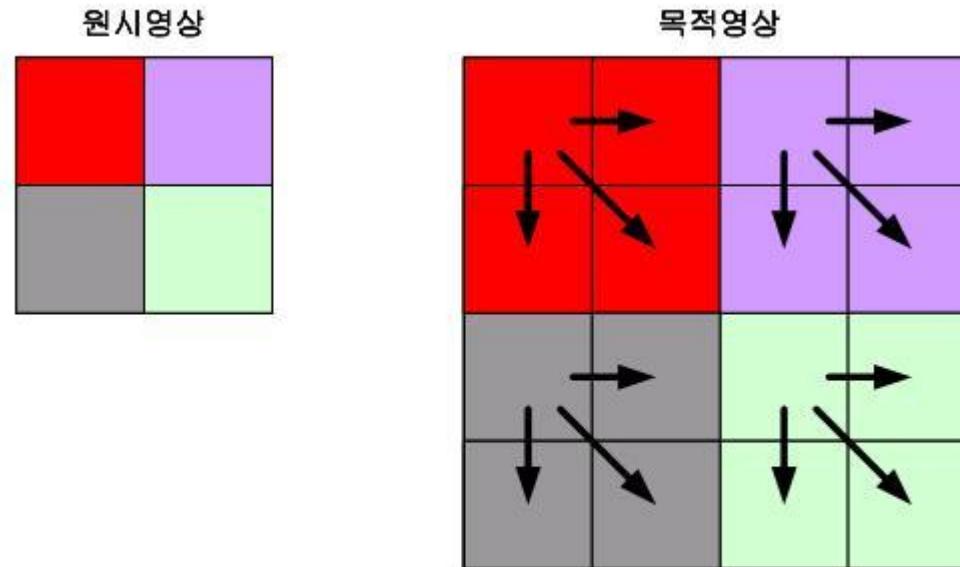
Nearest neighbor interpolation

- 계산한 위치에서 가장 가까운 원시 픽셀을 선택하는 방법
- (e.g.) $I(10,11) \rightarrow d(9.8,9.2) \approx d(10,9)=55$
- 처리 속도는 빠르지만 결과 영상의 질이 좋지 않음



가장 인접한 이웃 화소 보간법(Nearest Neighbor Interpolation)

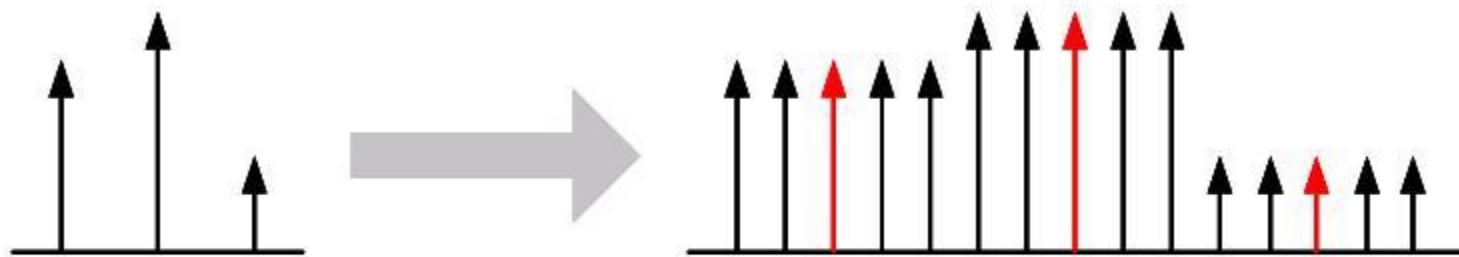
- 값을 할당받지 못한 목적 영상의 화소에서 가장 가깝게 이웃한 원시 화소의 값을 할당받은 목적 영상의 화소 값을 복사해서 사용하는 것



[그림 8-14] 가장 인접한 이웃 화소 보간법의 동작

가장 인접한 이웃 화소 보간법(계속)

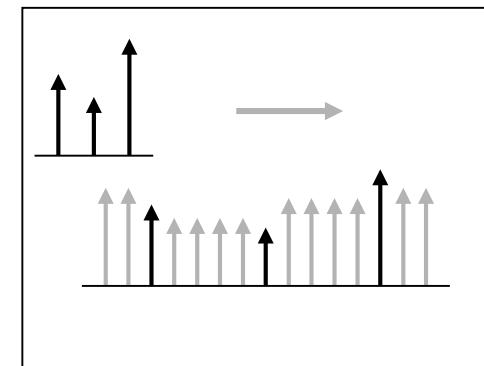
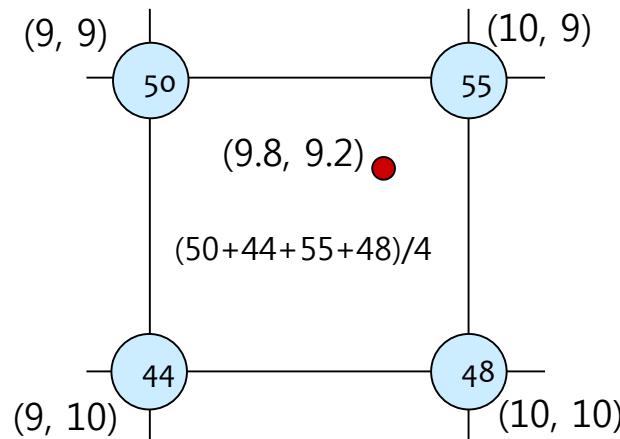
- 단순히 이웃 화소를 복사하여 사용하므로 처리 속도가 빠름
- 새로운 화소 값을 계산하지 않고 입력 화소 내에서만 찾기 때문에 원래의 영상과 전혀 다른 영상을 출력하는 오류가 발생
 - 하나의 입력 화소에 대응하는 출력 화소 수가 많을수록 영상의 질은 떨어지며, 영상 내에 톱니 모양이라고 하는 시각적인 뭉툭함(Blockiness)이 발생



[그림 8-16] 4배 보간된 영상에서 영상 품질 저하

Neighbor averaging interpolation

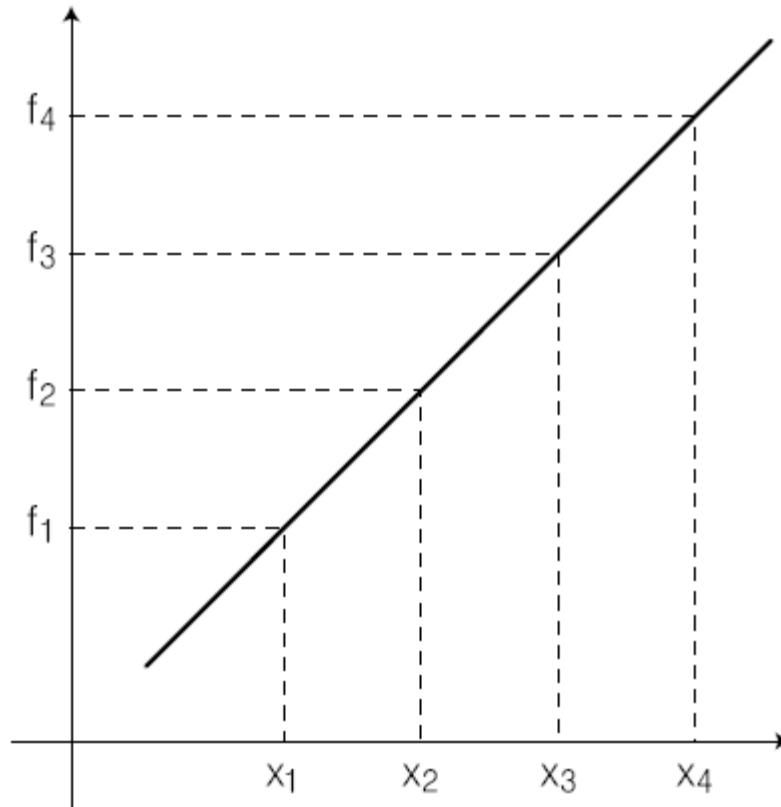
- Find a neighborhood average
- Two dimensionally using all four neighbors
- More computationally intensive but provide more visually pleasing result
- (e.g.) $I(10,11) \rightarrow d(9.8,9.2) = (50+44+55+48)/4 \approx 49$



4배 확대

선형 보간법 (Linear Interpolation)

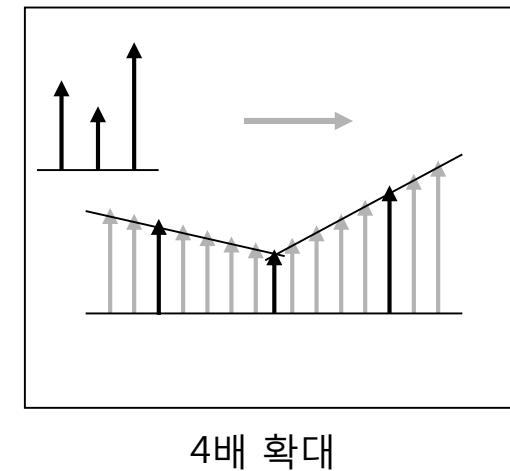
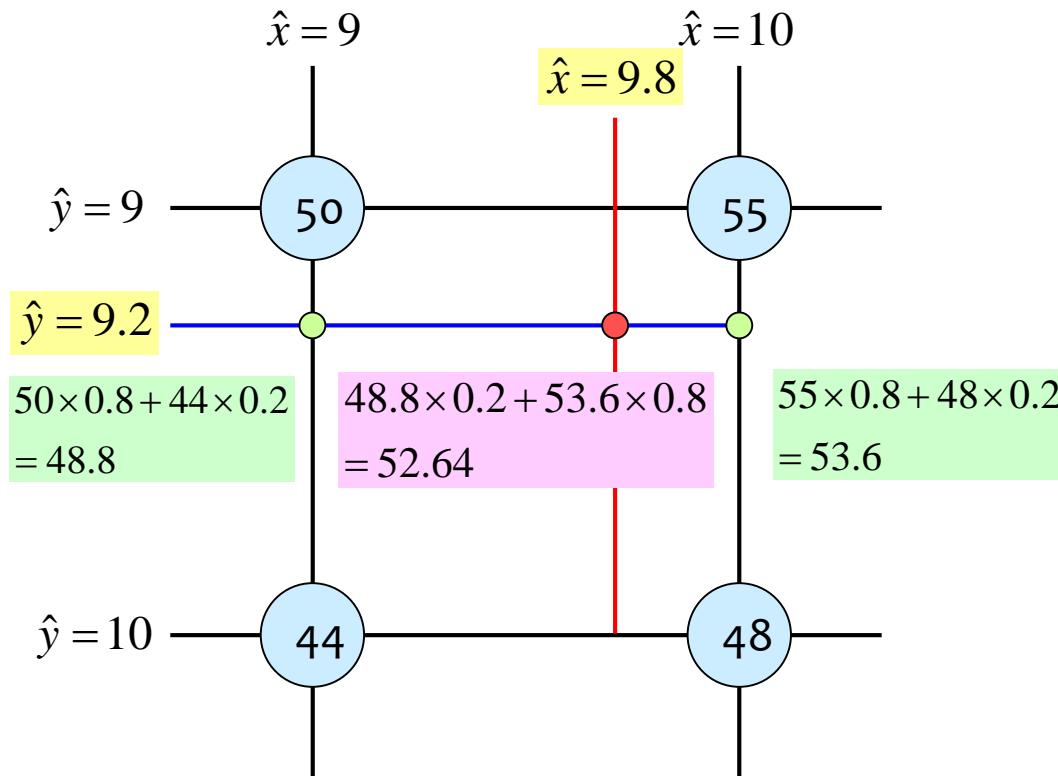
- 원시 영상의 화소 값 두 개를 이용하여 원하는 좌표에서 새로운 화소 값을 계산하는 간단한 방법



[그림 8-17] 선형 보간법의 원리

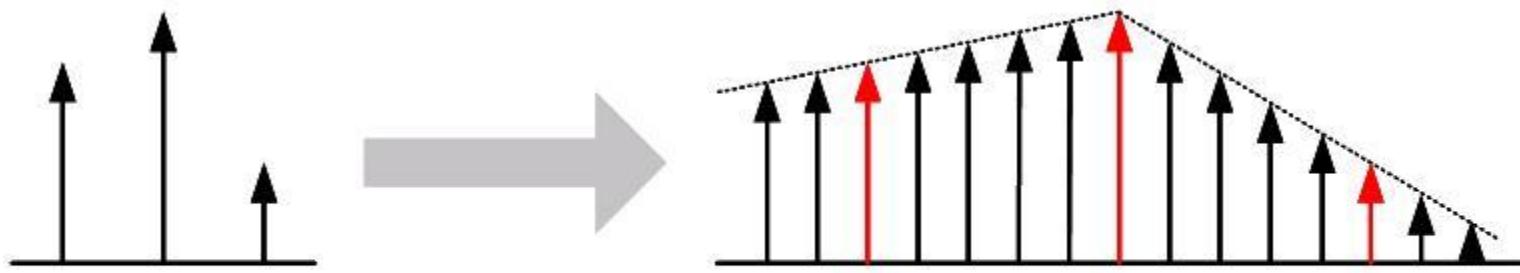
Bilinear interpolation

- 새로운 픽셀을 생성하기 위해 네 개의 가장 가까운 픽셀들에 가중치를 곱한 값들의 합을 사용
- 보다 자연스러운 영상을 산출



▶ 양선형 보간법의 장단점

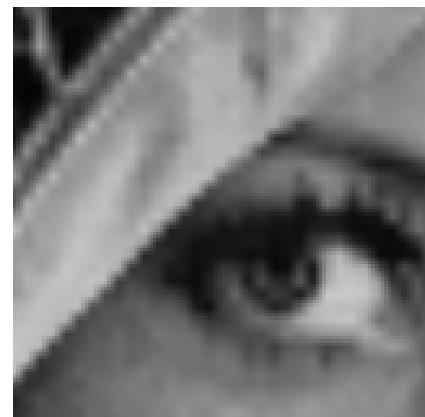
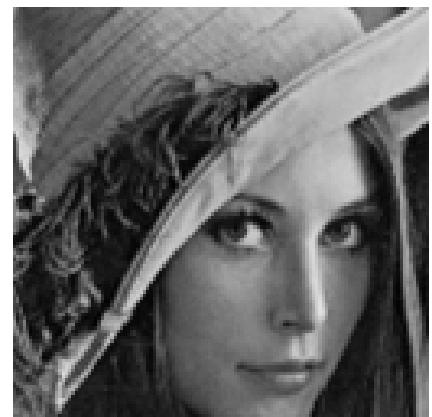
- 장점: 가장 인접한 화소 보간법에 비해 더 스무딩한 영상을 출력함.
- 단점: 화소당 선형 보간을 세 번씩 수행해야 하므로 상당히 많은 계산량이 소모됨.



[그림 8-20] 4배 확대된 양선형 보간



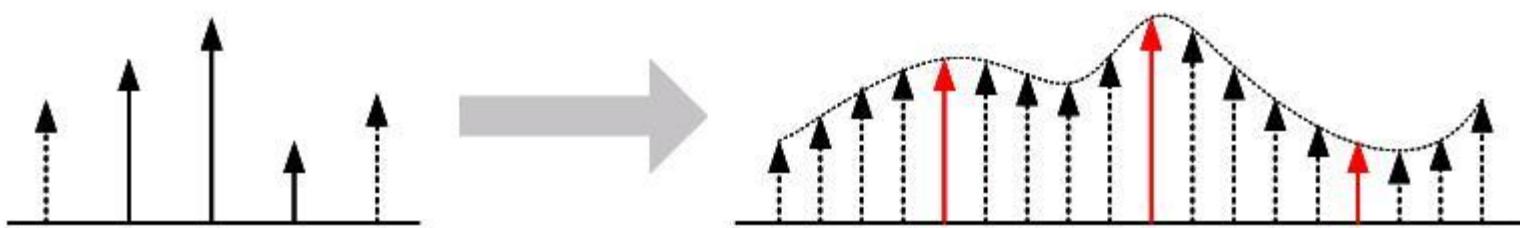
Nearest neighbor



Bilinear

고차 보간법

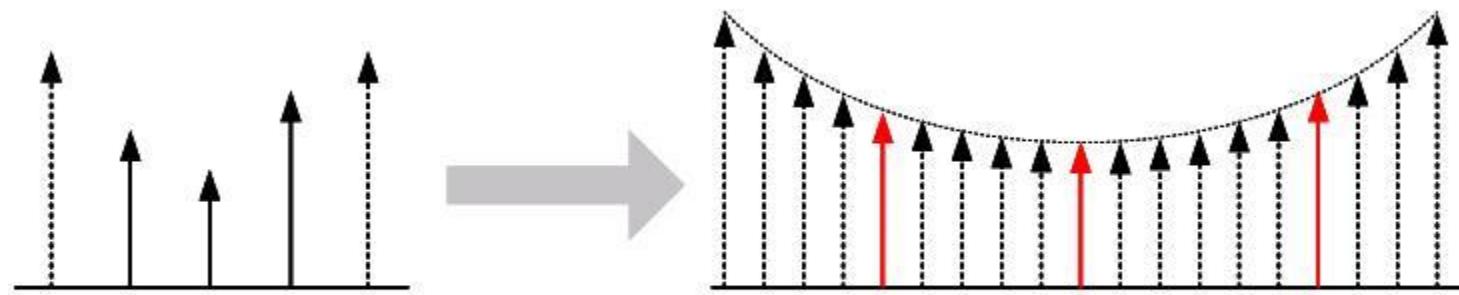
- 더 많은 이웃 화소를 참조하므로 값을 할당받지 못한 화소 값을 쉽게 추정할 수 있음.
- 3차 회선과 B-스플라인이 대표적
- 3차원 회선 보간법
 - 4×4의 이웃 화소를 참조하여 보간을 수행.
 - 양선형 보간법보다 더 많은 화소를 참조하므로 보간된 영상의 품질도 더 좋음.
 - 이웃 화소를 16개 참조하므로 계산 시간이 더 소요됨.



[그림 8-21] 3차원 회선 보간법으로 보간

▶ B-스플라인 보간법

- 이상적인 보간 함수는 저주파 통과 필터인데, B-스플라인 함수는 그중에서 도 상당히 좋은 저주파 통과 필터
→ B-스플라인 함수는 보간 함수 중에서 가장 스무딩한 영상 출력



[그림 8-22] B-스플라인 보간법으로 보간

Section 04 영상의 스케일링 기하학적 변환

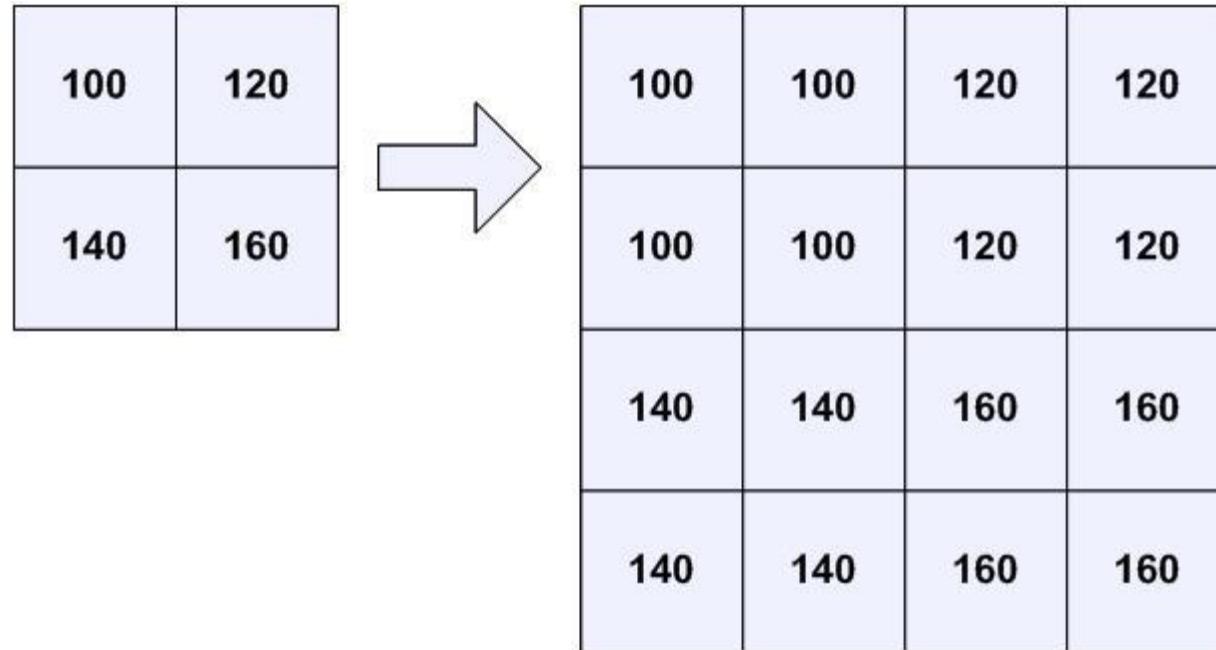
스케일링(Scaling)

- 디지털 영상의 모양은 변화시키지 않은 채 크기만을 확대하거나 축소하는 변환
 - 영상을 확대하는 것을 확대(Magnification), 스케일링 업(Scaling Up), 줌(Zooming), 업 샘플링(Up Sampling)이라고 하며,
 - 영상을 축소하는 것을 축소(Minification), 스케일링 다운(Scaling Dawn), 데시메이션(Decimation), 다운 샘플링(Dawn Sampling)이라 함.
- 스케일링 변환은 원 영상의 해상도를 떨어뜨리는 특징이 있어 결과 영상의 품질은 당연히 떨어짐.

영상의 확대 스케일링 변환

▣ 가장 인접한 이웃 화소 보간법을 이용한 영상 확대

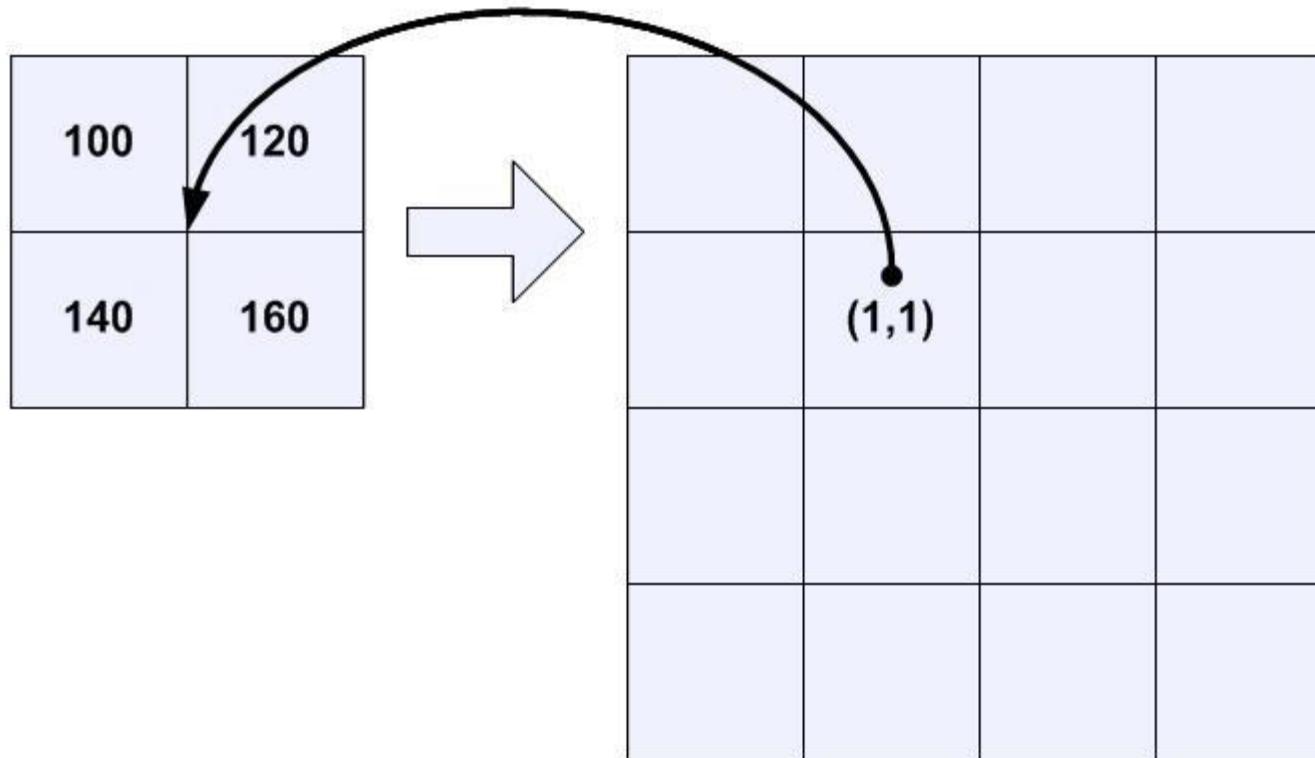
- 확대 배율만큼 화소를 반복적으로 복사해서 사용하므로 쉽고 빠르게 확대와 보간이 수행됨.



[그림 8-24] 영상을 2배 확대하여 가장 인접한 이웃 화소로 보간

영상의 확대 스케일링 변환[계속]

▶ 양선형 화소 보간법을 이용한 영상 확대

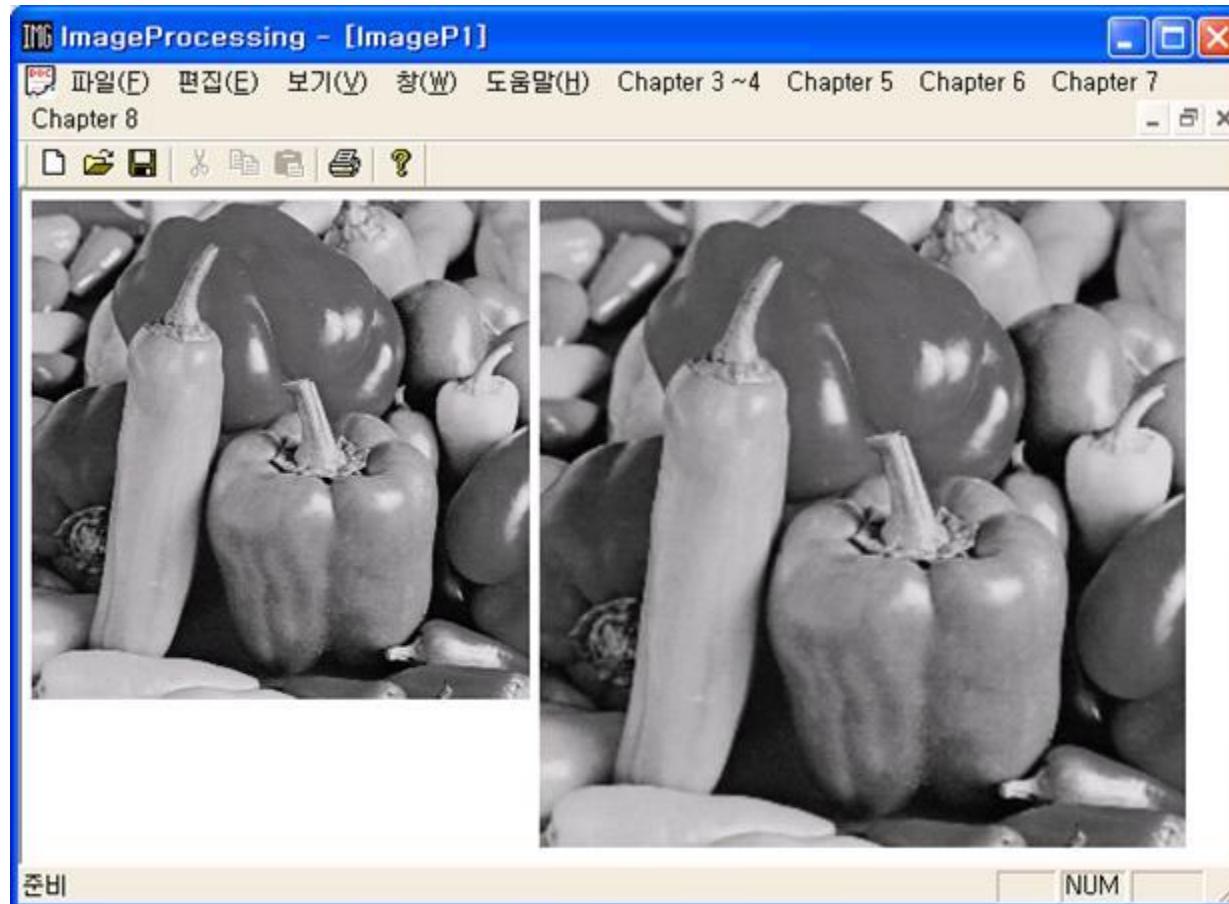


[그림 8-25] 양선형 보간법을 실제로 적용한 예

[실습하기 8-2] 양선형 보간법 이용한 영상 확대 프로그램

⑤ 프로그램 실행 결과 영상

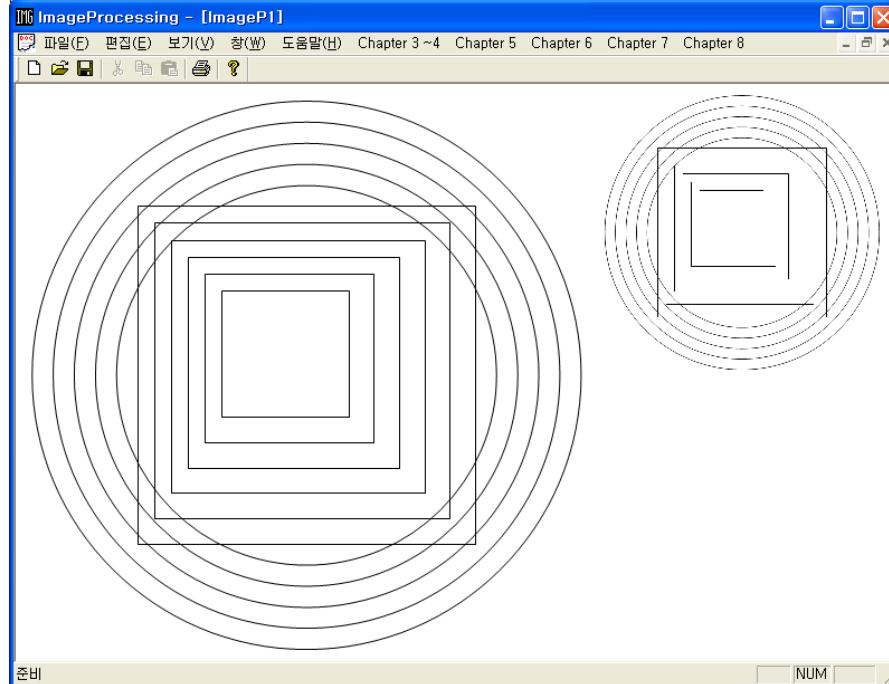
- 원시 영상을 2배로 확대한뒤 빈 화소에는 양선형 보간법을 적용한 것
- 결과 영상이 가장 인접한 이웃 화소 보간법의 결과보다 훨씬 부드러워짐.



양선형 보간이 수행된 결과 영상

영상의 축소 스케일링 변환

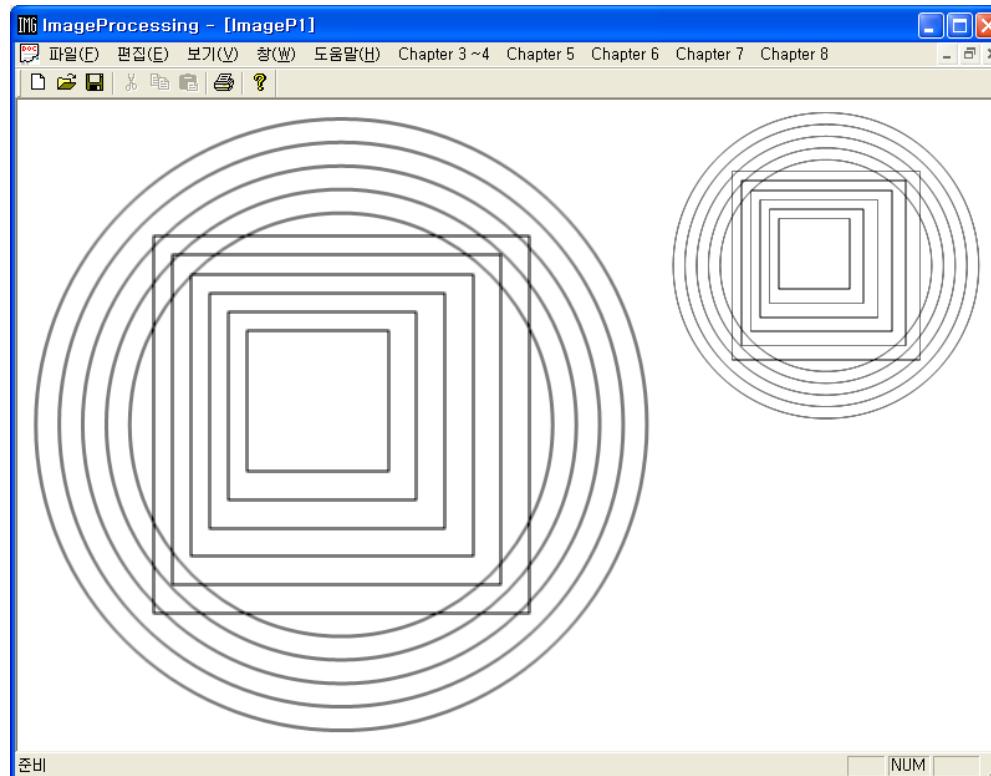
- ▶ 영상의 크기를 줄이는 영상의 축소 스케일링 변환식은 다음과 같이 확대와는 반대 개념
- ▶ 에일리어싱(Aliasing): 영상의 크기를 많이 축소하려고 너무 낮은 비율로 샘플링을 수행하면 화소 수를 너무 적게 취하게 되어 영상의 세부 내용을 상실하게 되는 현상



[그림 8-26] 서브 샘플링으로 세밀한 정보가 손실된 결과 영상

영상의 축소 스케일링 변환[계속]

- 서브 샘플링 과정에서 세부 내용을 잃어버리는 문제를 해결하려면 서브 샘플링을 수행하기 전에 먼저 영상의 블러링(Blurring)을 수행하면 됨.
- 즉, 저주파 통과 필터링을 통하여 블러링된 영상에서 서브 샘플링을 수행하면 세부 내용을 보존할 수 있음.

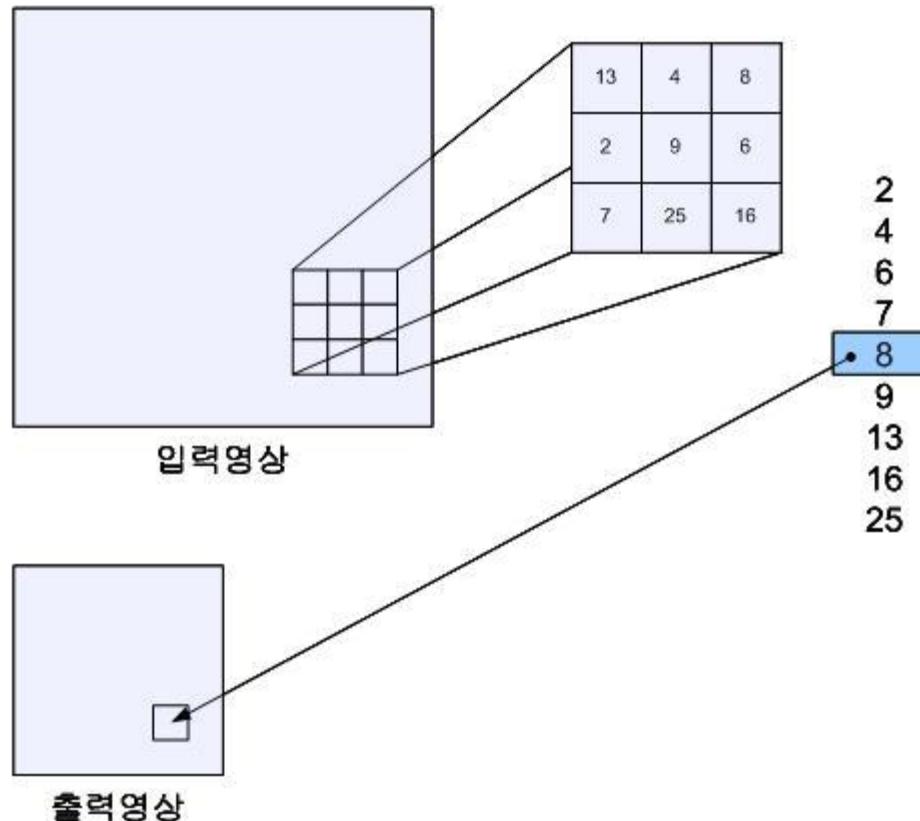


[그림 8-27] 블러링 처리한 뒤 서브 샘플링된 결과 영상

영상의 축소 스케일링 변환[계속]

▶ 미디언 표현

- 미디언(Median) 표현은 화소 블록을 중간(Median) 값으로 대치한 뒤 이 값을 샘플링하여 축소 영상의 화소로 사용하는 방법

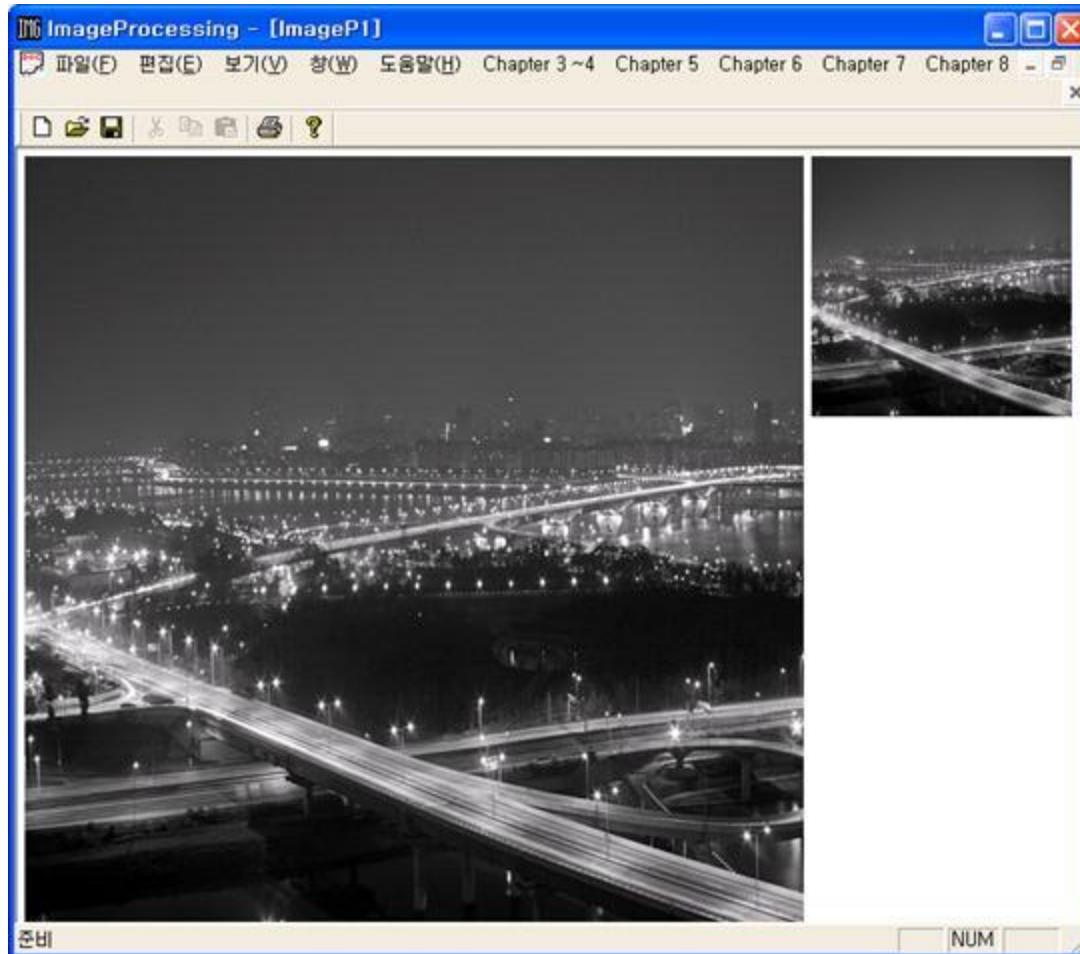


[그림 8-28] 미디언 표현을 이용한 서브 샘플링 동작 과정

[실습하기 8-3] 미디언 표현을 이용한 서브 샘플링 프로그램

⑤ 프로그램 실행 결과 영상

- 블러링 전처리로 서브 샘플링된 영상과는 다르게 원본 영상만큼 화질이 선명

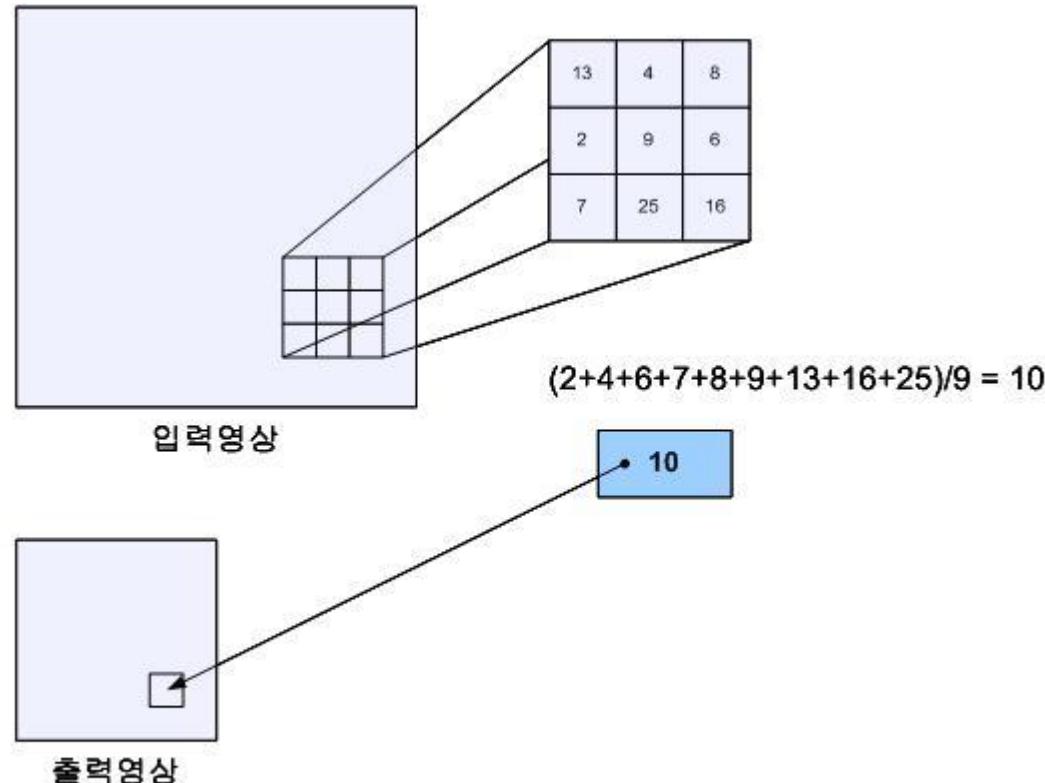


미디언 표현을 이용해 서브 샘플링된 결과 영상

영상의 축소 스케일링 변환[계속]

▶ 평균 표현

- ▶ 평균(Mean) 표현은 미디언 표현과 비슷하게 화소 블록을 블록 내 화소의 평균값으로 대치하는 방법
- ▶ 이렇게 얻은 평균값이 해당 축소 영상의 화소 값으로 사용됨.

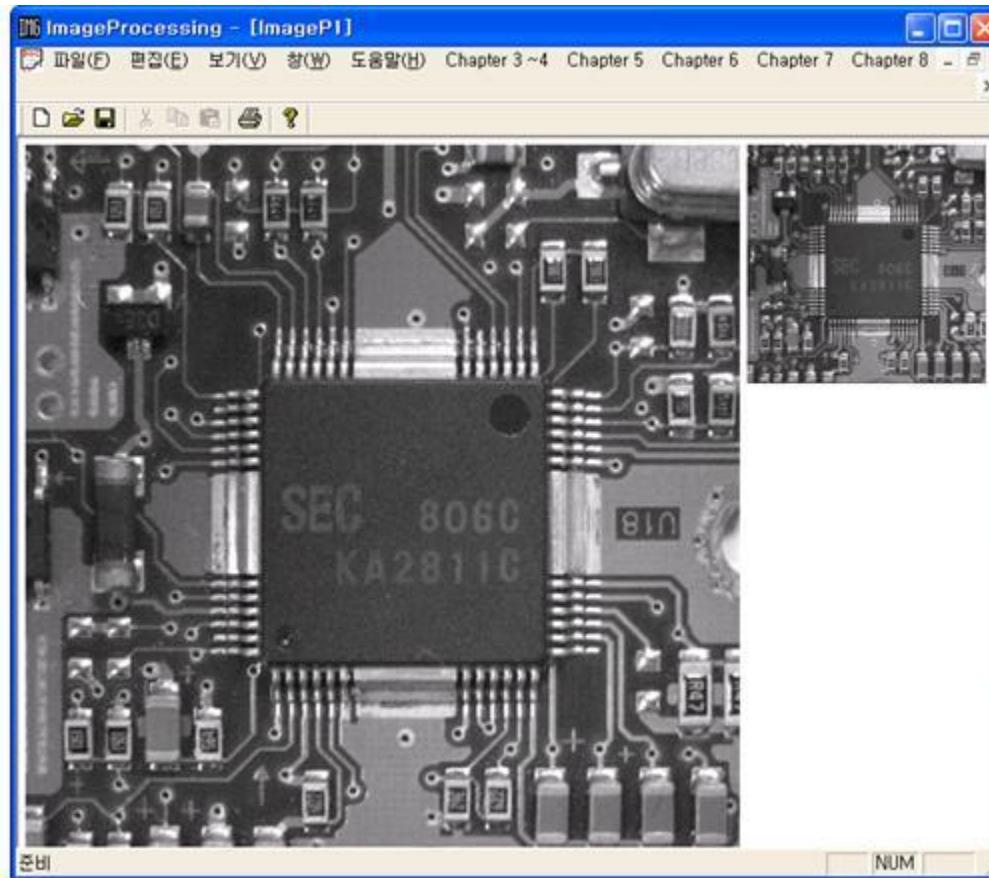


[그림 8-29] 평균 표현을 이용한 서브 샘플링의 동작

[실습하기 8-4] 평균 표현을 이용한 서브 샘플링 프로그램

⑤ 프로그램 실행 결과 영상

- 영상의 크기가 작아지고 스무딩해짐.



평균 표현을 이용해 서브 샘플링된 결과 영상

요약

▣ 기하학적 변환

- 영상을 구성하는 화소의 공간적 위치를 임의의 기하학적 변환으로 재배치하는 과정

▣ 선형 기하 연산

- 곡선이 전혀 없는 영상을 대상으로 평행이동, 회전, 스케일링 등 화소의 재배치를 수행

▣ 비선형 기하 처리

- 영상을 찌그러뜨리고 구부려서 곡선으로 처리하는 방법
- 워핑 변환과 모핑 변환이 대표적

▣ 사상

- 주어진 조건에서 현재의 데이터를 원하는 목표로 만드는 것

요약

▶ 전방향 사상

- 입력 영상의 모든 화소에서 출력 영상의 새로운 화소 위치를 계산하고, 입력 화소의 밝기 값을 출력 영상의 새로운 위치에 복사하는 방법
- 오버랩 문제와 홀 문제 발생
 - 오버랩 문제: 서로 다른 입력 화소 두 개를 똑같은 출력 화소에 사상하는 것. 새롭게 생성된 화소 값이 어떤 입력 화소에 근거하는지 정할 수 없으므로 처리된 결과 영상이 불명확
 - 홀 문제 : 전방향 사상에서 입력 영상의 화소가 목적 영상 내의 출력 화소에 없는 것. 입력 화소에 출력 화소 값이 배당되지 않으므로 정확한 영상처리를 할 수 없음.

▶ 역방향 사상

- 목적 영상의 화소를 조사하여 몇 가지 역변환으로 원시 영상의 화소를 구한 뒤 목적 영상의 화소 값을 생성하려고 사용
- 홀과 오버랩 문제가 발생하지 않아 기하학 처리에서 유용

요약

▣ 보간법

- 화소 값을 할당받지 못해 품질이 좋지 못한 것을 방지하기 위해 빈 화소에 값을 할당하여 좋은 품질의 영상을 만드는 방법

▣ 가장 인접한 이웃 화소 보간법

- 값을 할당받지 못한 목적 영상의 화소에서 가장 가깝게 이웃한 원시 화소의 값을 할당받은 목적 영상의 화소 값을 복사해서 사용하는 것
- 원시 화소에서 계산된 좌표가 정수가 아니면 가장 가까이에 있는 유효한 화소 좌표를 선택하는 것
- 처리 속도가 빠르나 하나의 입력 화소에 대응하는 출력 화소 수가 클수록 영상의 질은 떨어지며, 영상 내에 텁니 모양이라고 하는 시각적인 둥툭함이 발생

▣ 양선형 보간법

- 화소당 선형 보간을 세 번 수행, 새롭게 생성된 화소는 가장 가까운 화소 네 개에 가중치를 곱한 값을 합해서 얻음. 각 가중치는 각 화소에서의 거리에 정비례하도록 선형적으로 선택
- 가장 인접한 화소 보간법보다 더 스무딩한 영상을 출력하나 화소당 선형 보간을 세 번씩 수행해야 하므로 상당히 많은 계산량이 소모됨.

요약

▣ 고차 보간

- 더 많은 이웃 화소를 참조하므로 값을 할당받지 못한 화소 값을 쉽게 추정할 수 있음.

▣ 3차원 회선 보간법

- 4×4 의 이웃 화소를 참조하여 보간 수행
- 양선형 보간법보다 더 많은 화소를 참조하므로 보간된 영상의 품질이 더 좋으나 이웃 화소를 16개 참조하므로 계산 시간이 더 소요됨.

▣ B-스플라인 보간 함수

- 상당히 좋은 저주파 통과 필터로, 보간 함수 중에서 가장 스무딩한 영상을 출력

▣ 스케일링

- 디지털 영상의 모양은 변화시키지 않은 채 크기만을 확대하거나 축소하는 변환

요약

- Geometric Transforms

- 수식이나 변환 관계에 의해 픽셀들의 위치를 변경하는 변환

- Mapping by spatial transform

- 방식: forward 및 backward mapping
 - 종류: Affine transform 및 Warping (Perspective transform)

- Gray-level interpolation

- Nearest neighbor interpolation
 - Neighbor averaging interpolation
 - Bilinear interpolation

Reference

- R. Gonzalez, R. Woods, **Digital Image Processing (2nd Edition)**, Prentice Hall, 2002
- Scott E Umbaugh, **Computer Imaging**, CRC Press, 2005



Thank you